Haust 2016

Þýðendur
T-603-THYD

Homework 1

Nemandi:
Steinn Elliði Pétursson

Kennitala:
250594-2759

19. september 2016

Kennari:
Friðjón Guðjohnsen

1

Write Regular expressions describing the following languages over the alphabet $\Sigma = a, b$

a)

All non-empty strings that start and end with the same symbol.

Solution.

$$(a[ab]^*a)|(b[ab]^*b)|a|b$$

b)

All strings that do not have bbb as substring (ϵ included)

Solution.

$$\hat{}((?!bbb)[ab])^*\$$$

c)

All strings of even length that have aa as a substring.

Solution.

$$(ab|ba|bb|aa)^*(aa)^+(ab|ba|bb|aa)^*$$

d)

Problem. *All strings that contain at least one a and at least one b.*

Solution.

$$[ab]^*(ab|ba)[ab]^*$$

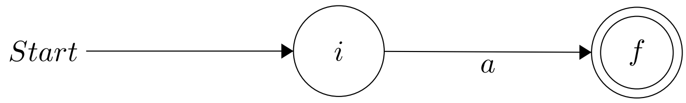
2

Use Algorithm 3.23 in the textbook (on page 159) to transform the regular expression $a((aa)^*|b)^*b$ into a nondeterministic finite automation (NFA). Draw the NFA and show all the intermediate steps in your construction.

Solution. *Following are the steps taken to generate an NFA from the above regular expression.*

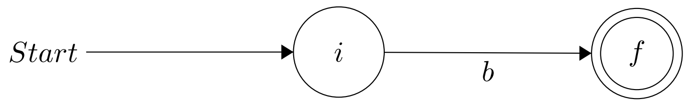
2.1

a



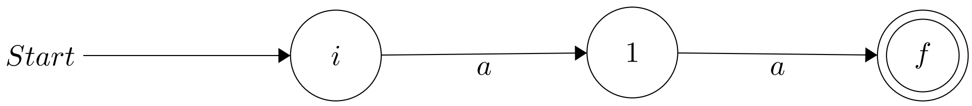
2.2

b



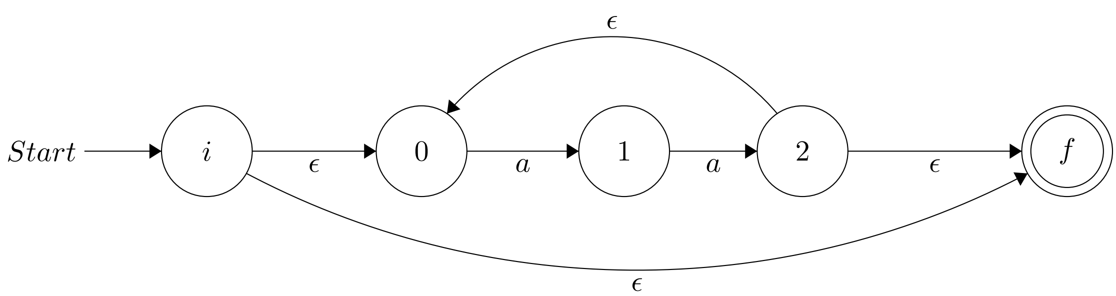
2.3

aa



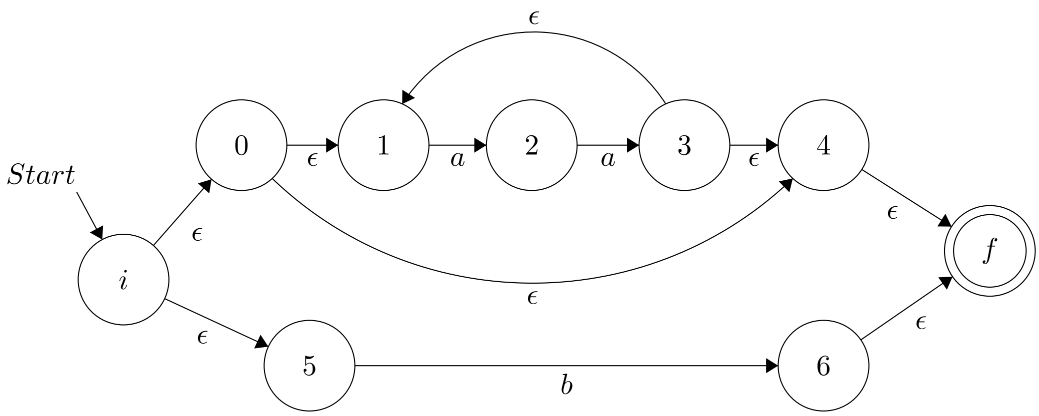
2.4

$(aa)^*$



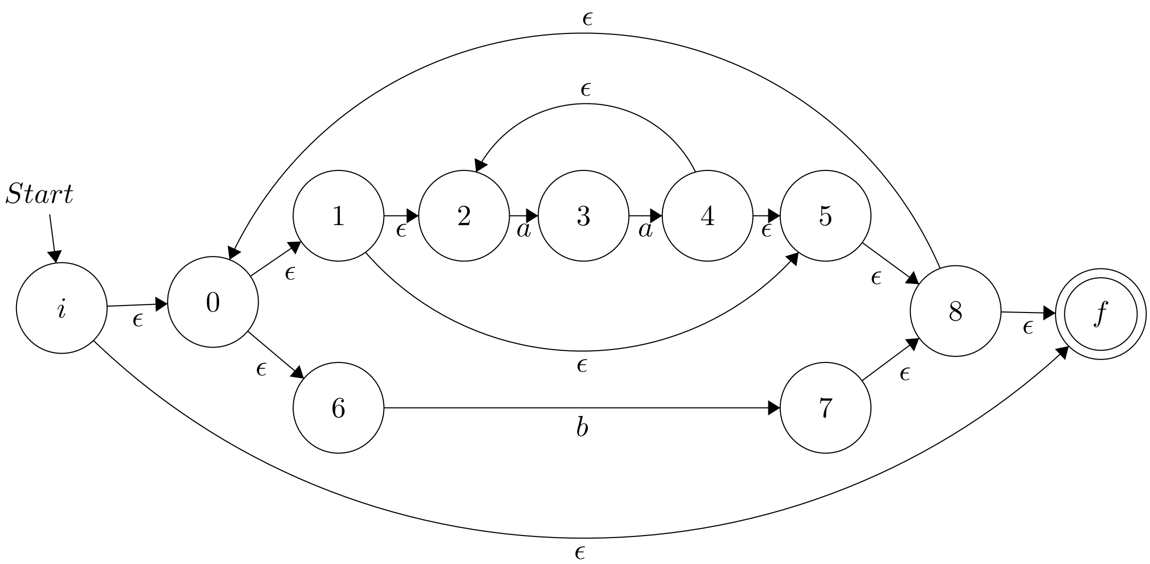
2.5

$(aa)^*|b$



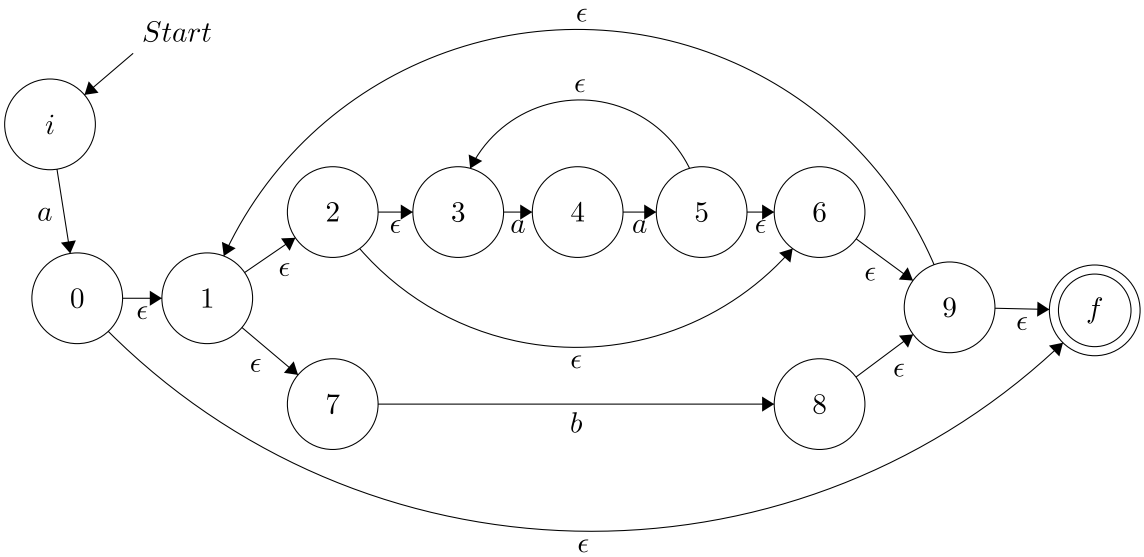
2.6

$((aa)^*|b)^*$



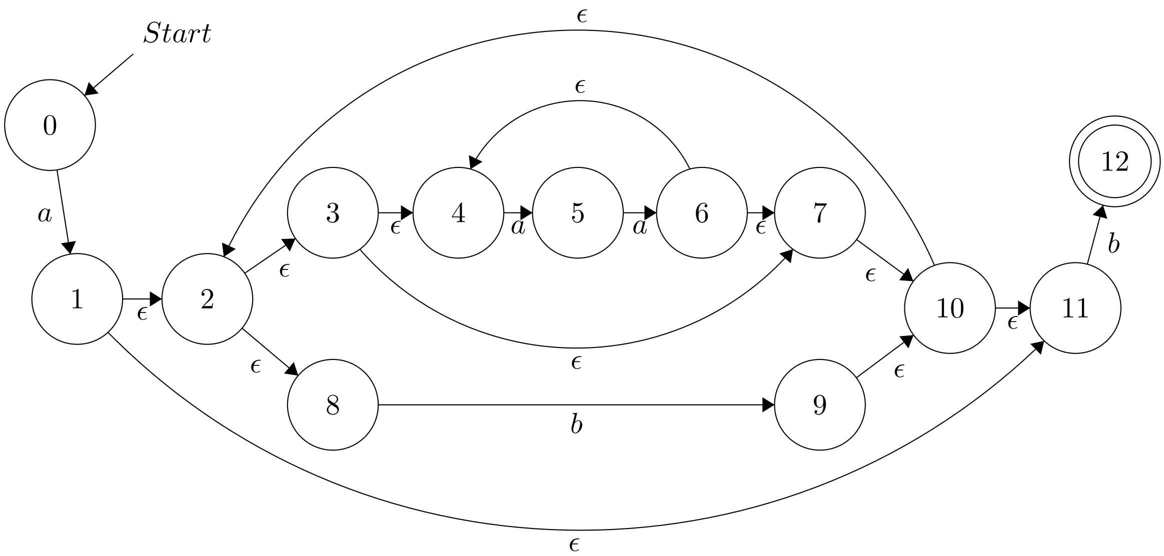
2.7

$a((aa)^*|b)^*$



2.8

$a((aa)^*|b)^*b$ The fully constructed NFA



3

Use algorithm 3.20 in the textbook (on page 153) to transform the NFA you constructed in Exercise 3 into a DFA. Draw the DFA and show all intermediate steps.

Solution.

Let's start by finding the start state of the DFA: $\epsilon - closure(0) = A = \{0\}$.
Next let's find the transitions for a and b:
$Dtran[A, a] = \epsilon - closure(move(A, a))$
$move(A, a) = \{1\}$
$\epsilon - closure(\{1\}) = \{1, 2, 3, 4, 7, 8, 10, 11\}$
Let us call this set B
$Dtran[A, b] = \epsilon - closure(move(A, b))$
$move(A, b) = \{\}$
$\epsilon - closure(\{\}) = \{\}$
This results in an invalid state which shall be called F. The invalid state can only redirect to itself since there is no way to end up with a valid solution if we end up in the invalid state.
After the above transition we have the following transition table.

NFA State	DFA State	a	b
{0}	A	B	F
{1,2,3,4,7,8,10,11}	B	?	?
{}	F	F	F

Now let us apply the algorithm on set B:
$Dtran[B, a] = \epsilon - closure(move(B, a))$
$move(B, a) = \{5\}$
$\epsilon - closure(\{5\}) = \{5\}$
This gives us $Dtran[B, a] = \{5\}$ let's call this set C
$Dtran[B, b] = \epsilon - closure(move(B, b))$
$move(B, b) = \{9, 12\}$
$\epsilon - closure(\{9, 12\}) = \{2, 3, 4, 7, 8, 9, 10, 11, 12\}$
This gives us $Dtran[B, a] = \{2, 3, 4, 7, 8, 9, 10, 11, 12\}$ let's call this set D.
Now we have the following transition table:

NFA State	DFA State	a	b
{0}	A	B	F
{1,2,3,4,7,8,10,11}	B	C	D
{5}	C	?	?
{2,3,4,7,8,9,10,11,12}	D	?	?
{}	F	F	F

Let's next apply the algorithm on set C:
$Dtran[C, a] = \epsilon - closure(move(C, a))$
$move(C, a) = \{6\}$
$\epsilon - closure(\{6\}) = \{2, 3, 4, 6, 7, 8, 10, 11\}$

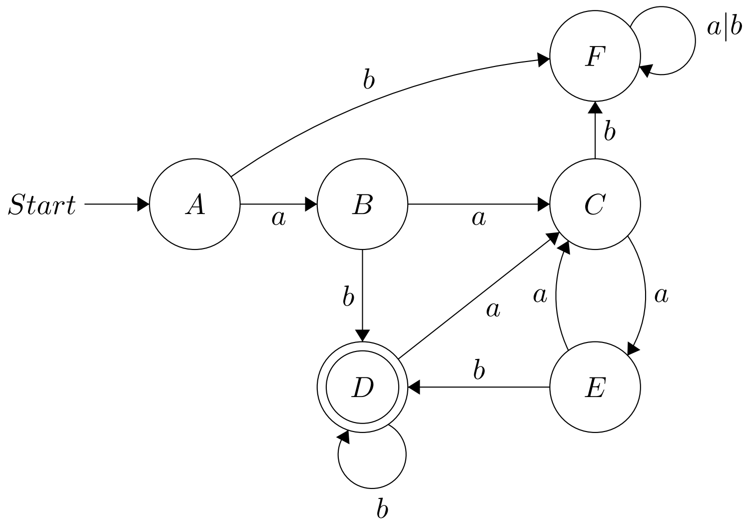
This gives us $Dtran[B, a] = \{2, 3, 4, 6, 7, 8, 10, 11\}$ let's call this set E
$Dtran[C, b] = \epsilon - closure(move(C, b))$
$move(C, b) = \{\}$
$\epsilon - closure(\{\}) = \{\}$
This results in the invalid state F
Now we have the following transition table:

NFA State	DFA State	a	b
{0}	A	B	F
{1,2,3,4,7,8,10,11}	B	C	D
{5}	C	E	F
{2,3,4,7,8,9,10,11,12}	D	?	?
{2,3,4,6,7,8,10,11}	E	?	?
{}	F	F	F

Applying the same methods on sets D and E we get the following results:
$Dtran[D, a] = \{5\}$ which is equal to the set C.
$Dtran[D, b] = \{2, 3, 4, 7, 8, 9, 10, 11, 12\}$ which is equal to D itself.
$Dtran[E, a] = \{5\}$ which is equal to the set C.
$Dtran[E, b] = \{2, 3, 4, 7, 8, 9, 10, 11, 12\}$ which is equal to D.
This gives us the complete transition table for the NFA:

NFA State	DFA State	a	b
{0}	A	B	F
{1,2,3,4,7,8,10,11}	B	C	D
{5}	C	E	F
{2,3,4,7,8,9,10,11,12}	D	C	D
{2,3,4,6,7,8,10,11}	E	C	D
{}	F	F	F

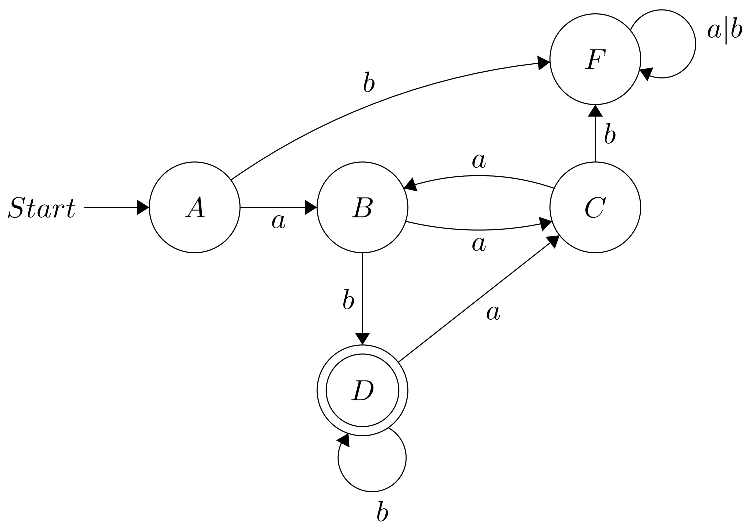
Following is the DFA constructed from the transition table:



4

Simplify the DFA created in the previous exercise into an equivalent DFA consisting of as few states as possible.

Solution.



5

Do the following pairs of regular expressions describe the same language?

a)

$\epsilon|a(a|b)^*b$ and $(a(a|b)^*b)^*$

Solution.

i) Let's call the former regular expression A (A $= \epsilon|a(a|b)^*b$) and the latter B (B $= (a(a|b)^*b)^*$)

ii) Let's begin by observing that both regular expressions accept ϵ. ϵ is a part of the definition of A but it is also allowed in B because of the kleene star.

iii) Now except for the empty string A has the only constraint that it accepts strings over the language $\{a, b\}$ which start with a and end with b.

iv) Observe that the exact same constraints happen to apply to B if we leave out the outer kleene star: $a(a|b)^*b$. So assuming that the outer kleene star will have the value 1 $(a(a|b)^*b\{1\})$ the pairs do describe the same language.

v) If the outer kleene star has a value > 1 then we will still have a regular expression which accepts strings over the language {a,b} that start with a and end with b since that is a concatenation of the case where the kleene star has a value of 1.

vi) Any additional constraints that might exist when the kleene star has a value > 1 are therefore irrelevant since although the concatenation might not accept all strings over the language {a,b} that start with a and end with b all the strings it does accept do start with a and end with b and are therefore a subset of the special case when the outer kleene star has a value of 1.

From i - vi above we can see that both regular expression do in fact describe the same language which is: The empty string or a string that starts with a and ends with b.

b)

a^* and $(aaa)^*(a|aa)$

Solution.

These regular expressions do not describe the same language since the former regular expression accepts ϵ but the latter does not.

6

Give a regular definition for the language consisting of all strings over the alphabet $\Sigma = \{a, b, c, d\}$ with no repeated letters.

Solution.

Let's begin by defining some regular expression which will help with the final definition.
$AInFront = a?b?c?d?|a?b?d?c?|a?c?b?d?|a?c?d?b?|a?d?b?c?|a?d?c?b?$
$BInFront = b?a?c?d?|b?a?d?c?|b?c?a?d?|b?c?d?a?|b?d?a?c?|b?d?c?a?$
$CInFront = c?b?a?d?|c?b?d?a?|c?a?b?d?|c?a?d?b?|c?d?b?a?|c?d?a?b?$
$DInFront = d?b?c?a?|d?b?a?c?|d?c?b?a?|d?c?a?b?|d?a?b?c?|d?a?c?b?$
With these definitions in place the final definition will be:
$NoRepeat = AInFront|BInFront|CInFront|DInFront$